

CLAIMS

What is claimed is:

- 1 1. A method, comprising:
2 processing a type-safe platform-independent firmware component during a
3 pre-boot phase of a computer platform; and
4 processing the type-safe platform-independent firmware component during an
5 operating system- (OS)-runtime phase.

- 1 2. The method of claim 1, wherein the type-safe platform-independent firmware
2 component is written in a type-safe intermediate language (IL) and is processed
3 using an interpreter for the IL during the pre-boot phase.

- 1 3. The method of claim 1, wherein the type-safe platform-independent firmware
2 component is written in a type-safe intermediate language (IL) and is processed
3 using a Just-in-Time (JIT) compiler for the IL during the pre-boot phase.

- 1 4. The method of claim 3, wherein processing the IL with the JIT compiler
2 generates an executable image, further comprising storing the executable image to
3 be accessible to platform firmware during a subsequent pre-boot phase for the
4 computer platform.

- 1 5. The method of claim 4, wherein the executable image is stored on a platform
2 firmware storage device.

1 6. The method of claim 4, wherein the executable image is stored in a portion of
2 a disk drive that is accessible to the platform firmware.

1 7. The method of claim 1, wherein the type-safe platform-independent firmware
2 component is written in a type-safe intermediate language (IL) and is processed
3 using a Just-in-Time (JIT) compiler for the IL during the OS-runtime phase.

1 8. The method of claim 1, wherein the type-safe platform-independent firmware
2 component is written in an intermediate language in accordance with the Common
3 Language Infrastructure (CLI) standard.

1 9. The method of claim 1, further comprising performing a type-safety
2 verification on the type-safe platform-independent firmware component prior to
3 processing it during the pre-boot phase.

1 10. The method of claim 1, further comprising performing a type-safety
2 verification on the type-safe platform-independent firmware component prior to
3 processing it during the OS-runtime phase.

1 11. A method comprising:
2 encoding source code corresponding to a firmware driver into a type-safe
3 intermediate language (IL)-encoded firmware driver; and
4 processing the type-safe IL-encoded firmware driver during a pre-boot phase
5 of a computer platform.

1 12. The method of claim 11, wherein the type-safe IL-encoded driver comprises a
2 PE/COFF (Portable Executable/Common Object File Format) image.

1 13. The method of claim 11, wherein the type-safe IL-encoded driver is encoded
2 into IL code defined by the Common Language Infrastructure (CLI) standard.

1 14. The method of claim 11, further comprising verifying the type-safe IL-encoded
2 firmware driver for type-safety prior to its processing.

1 15. The method of claim 11, wherein the type-safe IL-encoded firmware driver is
2 processed using an IL interpreter.

1 16. The method of claim 11, wherein the type-safe IL-encoded firmware driver is
2 processed using a Just-in-Time (JIT) compiler.

1 17. The method of claim 11, wherein the type-safe IL-encoded firmware driver is
2 compliant with the Extensible Firmware Interface (EFI) standard.

1 18. The method of claim 11, wherein the type-safe IL-encoded firmware driver is
2 accessed during OS-runtime by an operating system user mode.

1 19. The method of claim 11, wherein the type-safe IL-encoded firmware driver is
2 accessed during OS-runtime by an operating system kernel mode.

1 20. A machine-readable media to provide instructions, which when executed
2 perform operations comprising:
3 loading a type-safe processor-neutral firmware module into a pre-boot
4 environment; and

5 processing the type-safe processor-neutral firmware module via a virtual
6 processor in the pre-boot environment.

1 21. The machine-readable media of claim 20, wherein the instructions including
2 native instructions corresponding to an interpreter to operate as the virtual
3 processor. .

1 22. The machine-readable media of claim 20, wherein the instructions including
2 native instructions corresponding to a Just-in-Time (JIT) compiler to operate as the
3 virtual processor.

1 23. The machine-readable media of claim 20, wherein the instructions include
2 instructions written in intermediate language (IL) code.

1 24. The machine-readable media of claim 23, wherein the IL code is compliant
2 with the Common Language Infrastructure (CLI) standard

1 25. The machine-readable media of claim 20, wherein the media comprises a
2 firmware storage device, and the instructions comprise firmware instructions.

1 26. The machine-readable media of claim 20, wherein execution of the
2 instructions performs the further operation of publishing an interface via which an
3 operating system (OS) may access the type-safe processor-neutral firmware module
4 during OS runtime operations.

1 27. A computer system, comprising:
2 a system processor,

3 memory, coupled to the processor; and
4 a flash device on which firmware is stored, said firmware including a plurality
5 of firmware drivers written in a type-safe intermediate language and native
6 instructions comprising a virtual processor to be hosted by the system processor,
7 said virtual processor to process the plurality of firmware drivers during a pre-boot
8 phase for the computer system.

1 28. The computer system of claim 27, wherein the type-safe intermediate
2 language is compliant with the Common Language Infrastructure (CLI) standard.

1 29. The computer system of claim 28, wherein the virtual processor comprises a
2 CLI-compliant interpreter.

1 30. The computer system of claim 28, wherein the virtual processor comprises a
2 CLI-compliant Just-in-Time (JIT) compiler.